



# Putting your users in a Box

Greg Thain  
Condor Week 2013

# Outline

- › Why put job in a box?
- › Old boxes that work everywhere\*
  - » \*Everywhere that isn't Windows
- › New shiny boxes

# 3 Protections

- 1) Protect the machine from the job.
- 2) Protect the job from the machine.
- 3) Protect one job from another.

# The perfect box

- › Allows nesting
- › Need not require root
- › Can't be broken out of
- › Portable to all OSes
- › Allows full management:
  - Creation // Destruction
  - Monitoring
  - Limiting



# A Job ain't nothing but work

- › Resources a job can (ab)use
  - CPU
  - Memory
  - Disk
  - Signals
  - Network.



# Previous Solutions

## › HTCondor Preempt expression

- PREEMPT =

- TARGET.MemoryUsage > threshold

- ProportionalSetSizeKb > threshold

## › setrlimit call

- USER\_JOB\_WRAPPER

- STARTER\_RLIMIT\_AS

# From here on out...

- › Newish stuff

# The Big Hammer

- › Some people see this problem, and say
- › “I know, we’ll use a Virtual Machine”





# Problems with VMs

- › Might need hypervisor installed
  - The right hypervisor (the right Version...)
- › Need to keep full OS image maintained
- › Difficult to debug
- › Hard to federate
  
- › Just too heavyweight



# Containers, not VMs

- › Want opaque box
- › Much LXC work applicable here
- › Work with Best feature of HTCondor ever?

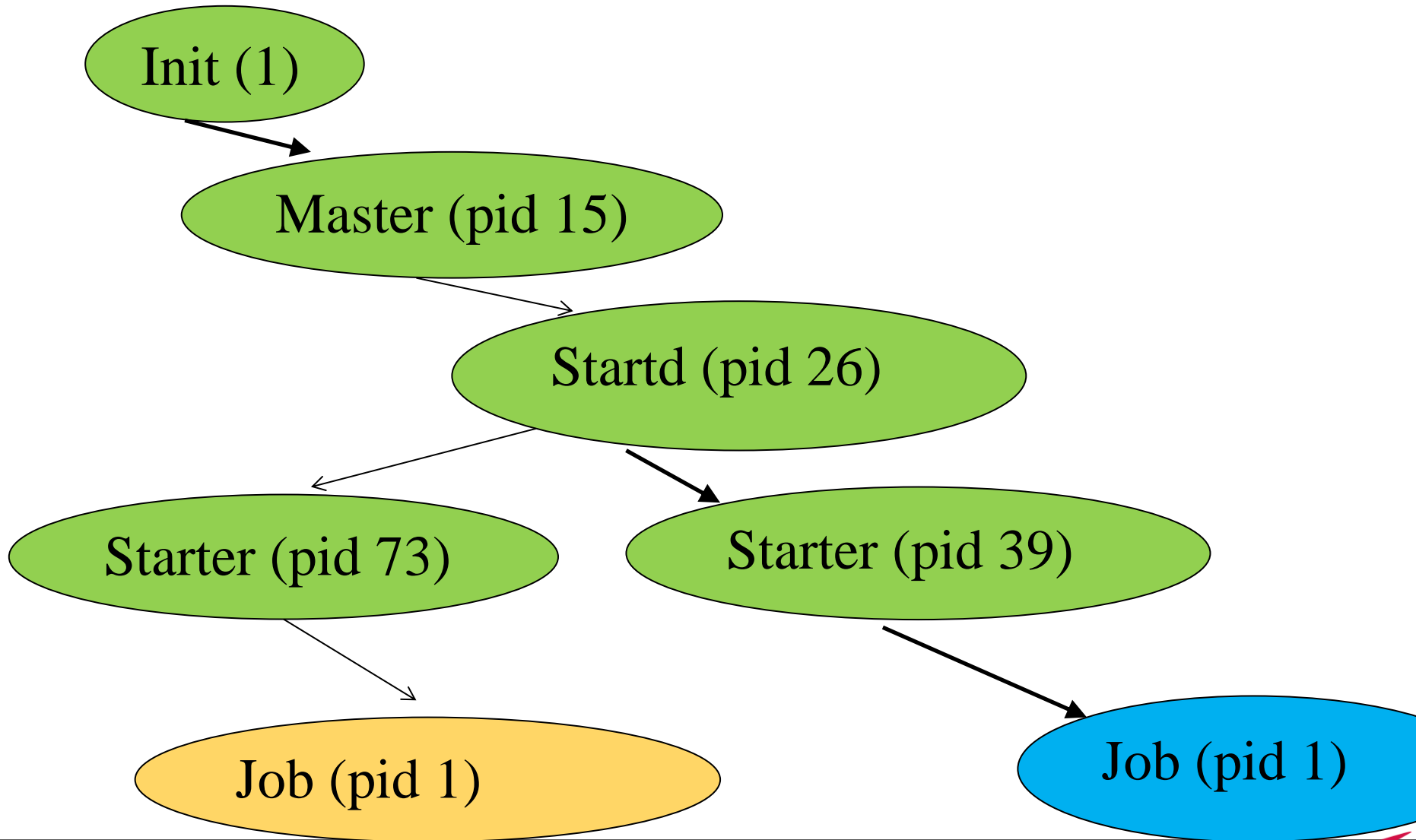
# CPU AFFINITY

- › `ASSIGN_CPU_AFFINITY=true`
- › Now works with dynamic slots
- › Need not be root
- › Any Linux version
  - Only limits the job

# PID namespaces

- › You can't kill what you can't see
- › Requirements:
  - HTCondor 7.9.4+
  - RHEL 6
  - `USE_PID_NAMESPACES = true`
    - (off by default)
  - Doesn't work with `privsep`
  - Must be root

# PID Namespaces



# Named Chroots

- › “Lock the kids in their room”
- › Startd advertises set
  - › `NAMED_CHROOT = /foo/R1, /foo/R2`
  - › Job picks one:
    - › `+RequestedChroot = “/foo/R1”`
- › Make sure path is secure!

# Control Groups aka “cgroups”

- › Two basic kernel abstractions:
  - › 1) nested groups of processes
  - › 2) “controllers” which limit resources

# Control Cgroup setup

- › Implemented as filesystem
  - Mounted on /sys/fs/cgroup, or /cgroup or ...
- › User-space tools in flux
  - Systemd
  - Cgservice
- › /proc/self/cgroup



# Cgroup controllers

- › Cpu
- › Memory
- › freezer

# Enabling cgroups

## › Requires:

- RHEL6
- HTCondor 7.9.5+
- Rootly condor
- No privsep
- `BASE_CGROUP=htcondor`
  
- And... cgroup fs mounted...

# Cgroups

- › Starter puts each job into own cgroup
  - Named `exec_dir + job id`
- › Procd monitors
  - Procd freezes and kills atomically
- › MEMORY attr into memory controller
- › CGROUP\_MEMORY\_LIMIT\_POLICY
  - Hard or soft
  - Job goes on hold with specific message

# Cgroup artifacts

## StarterLog:

```
04/22/13 11:39:08 Requesting cgroup
    htcondor/condor_exec_slot1@localhost for job
```

...

## ProcLog

```
cgroup to htcondor/condor_exec_slot1@localhost for ProcFamily
... 2727.
04/22/13 11:39:13 : PROC_FAMILY_GET_USAGE
04/22/13 11:39:13 : gathering usage data for family with root
pid 2724
04/22/13 11:39:17 : PROC_FAMILY_GET_USAGE
04/22/13 11:39:17 : gathering usage
```

```
$ condor_q
```

```
-- Submitter: localhost : <127.0.0.1:58873> : localhost
```

```
  ID          OWNER          SUBMITTED RUN_TIME ST PRI SIZE  
CMD  
  2.0      gthain              4/22 11:36 0+00:00:02 R 0 0.0 sleep 3600
```

```
>$ ps ax | grep 3600
```

```
gthain  2727  4268 4880 condor_exec.exe 3600
```

# A process with Cgroups

```
$ cat /proc/2727/cgroup
```

```
3:freezer:/htcondor/condor_exec_slot1@localhost
```

```
2:memory:/htcondor/condor_exec_slot1@localhost
```

```
1:cpuacct,cpu:/htcondor/condor_exec_slot1@localhost
```

```
$ cd  
/sys/fs/cgroup/memory/htcondor/condor_exec_sl  
ot1@localhost/  
$ cat memory.usage_in_bytes  
258048  
$ cat tasks  
2727
```

# MOUNT\_UNDER\_SCRATCH

- › Or, “Shared subtrees”
- › Goal: protect /tmp from shared jobs
- › Requires
  - Condor 7.9.4+
  - RHEL 5
  - Doesn't work with privsep
  - HTCondor must be running as root
  - MOUNT\_UNDER\_SCRATCH = /tmp,/var/tmp



# MOUNT\_UNDER\_SCRATCH

```
MOUNT_UNDER_SCRATCH=/tmp, /var/tmp
```

Each job sees private /tmp, /var/tmp

Downsides:

No sharing of files in /tmp

# Future work

- › Per job FUSE and other mounts?
- › non-root namespaces

# Not covered in this talk

- › Prevent jobs from messing with everyone on the network:
- › See Lark and SDN talks Thursday at 11

# Conclusion

- › Questions?
- › See cgroup reference material in kernel doc
  - <https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt>
- › LKN article about shared subtree mounts:
  - <http://lwn.net/Articles/159077/>